Práctica 3

Arboles Binarios de Busqueda y AVLs

Parte 1: Arboles binarios de búsqueda

1. Muestre en papel como se va armando un árbol binario de búsqueda (inicialmente vacío) con las siguientes operaciones:
   - Inserción de los siguientes elementos: 3, 1, 4, 6, 8, 2, 5, 7.

2. Muestre en papel como se va armando un árbol binario de búsqueda (inicialmente vacío) con las siguientes operaciones:
   - Inserción de los siguientes elementos: 15, 11, 9, 7, 5, 4, 1.

3. ¿Qué puede concluir sobre la altura del árbol a partir de los casos anteriores?

4. En base a la implementación de árboles binarios de búsqueda proporcionada por la cátedra, complete la misma agregando los siguientes métodos:
   - agregar(elemento) # Agrega elemento al árbol binario de búsqueda
   - incluye(elemento) # Retorna True si elemento está en el árbol, False en caso contrario
   - inorden() # imprime un recorrido inorden del árbol binario de búsqueda
   - preorden() # imprime un recorrido preorden del árbol binario de búsqueda
   - postorden() # imprime un recorrido postorden del árbol binario de búsqueda

5. Pruebe la implementación de árbol binario de búsqueda, armando un árbol como los de los ejercicios 1 y 2 para luego imprimirlos con los distintos recorridos (inorden, postorden y preorden). Compare con lo que previamente usted determinó que debería ser el resultado esperado.

Ejercicio 2: Arboles AVLs

1. Muestre en papel como se va armando un árbol AVL, (inicialmente vacío) con las siguientes operaciones:
   - Inserción de los siguientes elementos: 40, 20, 30, 38, 33, 36, 34, 37. Indique en caso que el árbol se des-balancee, el nodo que se des-balanceó y el tipo de rotación necesaria para balancearlo.

2. En base a la implementación de árboles AVL proporcionada por la cátedra, complete la misma agregando los siguientes métodos:
   - agregar(elemento) # Agrega elemento al árbol AVL
   - rotacionSimpleDerecha() # Realiza una rotación simple hacia la derecha del árbol que recibe el mensaje
   - rotacionSimpleIzquierda() # Realiza una rotación simple hacia la izquierda del árbol que recibe el mensaje
   - rotacionDobleDerecha() # Realiza una rotación Doble hacia la derecha del árbol que recibe el mensaje
   - rotacionDobleIzquierda() # Realiza una rotación doble hacia la izquierda del árbol que recibe el mensaje

   # los siguientes métodos deberían ser igual que los implementados para árboles binarios de búsqueda
   - incluye(elemento) # Retorna True si elemento está en el árbol, False en caso contrario
   - inorden() # imprime un recorrido inorden del árbol binario de búsqueda
   - preorden() # imprime un recorrido preorden del árbol binario de búsqueda
   - postorden() # imprime un recorrido postorden del árbol binario de búsqueda

Ejercicio 3

Implemente la clase Iterador que permite recorrer el contenido del árbol en orden. La misma tiene los métodos hasNext():bool y next():string que permiten acceder al contenido del árbol siguiendo el orden de los elementos.

A continuación se muestra un método que imprime el contenido de un árbol binario de búsqueda de strings referenciado por la variable abb:

```
Iterador it = new Iterador(abb);
while (it.hasNext()) {
	string valor = it.next();
	Console.Write(valor);
}
```

Ejercicio 4

El buque Manuel Belgrano del Instituto Argentino de Oceanográfico desarrolla desde hace más de un lustro una investigación oceanográfia cuyo objetivo es analizar las masas de agua que entran y salen del lecho oceánico argentino. En estos últimos cinco años, ha recogido diversas muestras del mismo punto geográfico, al menos una quincena de veces. En este marco, el buque al tomar una nueva muestra de agua física, la cual se representa como un número real, es inmediatamente almacenada en una subclase de **ArbolAVL** llamada **ArbolAVLDeMuestras**.

Su trabajo toma lugar luego de ser almacenada en la estructura de datos la nueva muestra, el cual consiste en calcular el mínimo delta histórico respecto a la nueva medición. El delta histórico es el resultado de realizar la diferencia entre la muestra más alta de las más bajas respecto a la nueva muestra y la más baja entre las más altas de la nueva muestra.

Usted debe implementar un método llamado minimoDeltaHistorico en la clase **ArbolAVLDeMuestras**, considerando que recibe como parámetro un número que representa la nueva muestra y debe devolver el mínimo delta histórico, según el criterio descripto anteriormente. En el caso que sea imposible encontrar alguno de estos dos valores necesarios para calcular el mínimo delta histórico debe devolver un valor igual a 0 para el menor y 999 para la mayor. (0 y 999 no son valores válidos, no están en el árbol).